PerAdaMagazine
towards pervasive adaptation
10.2417/2200903.1534

Designing and controlling trustworthy self-organising systems

Giovanna Di Marzo Serugendo and John Fitzgerald

Decision-making and adaptation based on dynamic policy enforcement promote predictability and control.

A self-organising system can arrange itself and modify its behaviour without receiving specific instructions to do so. Such systems are common in nature: flocks of birds respond to wind changes and colonies of ants structure themselves in response to a threat. But self-organisation is seen not only in nature. Increasingly, artificial systems such as robots, mobile networks and software services are able to self-organise, enabled by modern computing and network technologies. Such systems show some of the adaptability of their natural counterparts, but their behaviour is hard to control (to stop, reset or guide) and even harder to predict. So, can such systems be trusted? The challenge in our work is to provide means of designing and controlling artificial self-organising systems so that there is enough evidence to justify relying on them to perform safely, correctly and efficiently: and do this despite erratic behaviour by the environment or faulty components.

System dependability is a well-established field of study.[1] The main techniques for achieving and demonstrating dependability are to avoid the introduction of defects during design, to use over-engineering to tolerate faults should they arise anyway and to detect any remaining faults through system verification. When these techniques are applied, evidence is produced that can form the basis of a system's dependability argument. However, most of these methods assume a static system structure fixed during design, while real self-organising systems are dynamic, with components and agents joining and leaving, changing goals and reacting to events. Specific approaches targeting self-* systems vary from multi-layer reference architectures for self-adaptive systems[2] to analysis guidelines and specific agent-based solutions.[3] They do not address trustworthiness and controllability. To bridge this gap, we have been working on a software architecture and development method that allows mechanisms that both ensure and constrain the run-time behaviour of a self-organising system to be defined and analyzed at design-time. This therefore provides some assurance of its self-* capabilities.

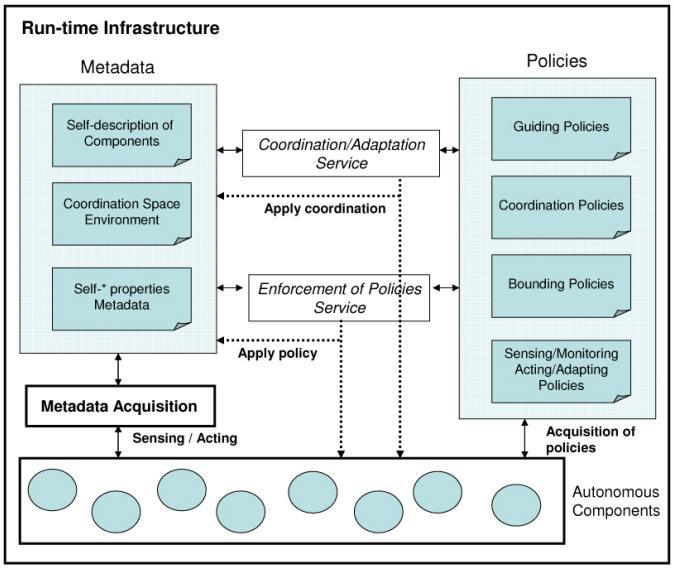

Figure 1. *The proposed architecture involves loosely coupled components, metadata and policies.*

We view the self-organising system as a collection of loosely coupled autonomous components. We gather and maintain metadata that describes components' functional specifications and non-functional characteristics such as availability levels and environment-related metadata (e.g. artificial pheromones). The system's behaviour (for example, reconfiguration to compensate for component failure) is governed by policies that describe the response of system components to detected conditions and changes in the metadata. When the system is live, both the components and the run-time infrastructure exploit metadata to support decision-making and adaptation in accordance with the policies.[4]

Continued on next page

To realise this approach, we have proposed a system architecture (see Figure 1) that involves autonomous components, repositories of metadata and executable policies, and reasoning services that dynamically enforce the policies on the basis of metadata values.[5] Metadata may be stored, published and updated at run-time by the run-time infrastructure and by the components themselves, both of which can also access policies at run-time.

Guiding policies are high-level goals (e.g. starting or stopping a swarm of robots); bounding policies define environmental limitations; sensing/monitoring policies define reflex behaviour for the components (e.g. if a metadata value reaches a threshold, an action must be taken). Policies may be generic, e.g. replacing a current (slow) component with a higher-performance equivalent. By accessing metadata about current performance, the reasoning engine can determine which of the available components must replace the failing one. In principle, policies can change dynamically, although allowing unconstrained change can affect dependability!

We have defined a development method in which the requirement and analysis phase identifies the functionality of the system along with self-* requirements specifying where and when self-organisation is needed or desired. A design phase determines the design of autonomous components (services, agents, etc.) and the mechanisms governing the their interactions and behaviour (e.g. trust, gossip, or stigmergy: indirect coordination through changes in the environment), addressing the self-* requirements. The implementation phase produces the run-time infrastructure.[5]

We have applied our approach in two case studies. First, we have developed dynamically resilient Web services where a client requesting a service specifies its choice of dependability at run-time, e.g. the Web service with the best dependability metadata is selected as the primary service, and others are used as alternatives if that one fails.[6] Second, we have designed self-organising robotic assembly systems. In response to an incoming product order, robotic modules self-organise—select each other and re-program themselves—to form an ad hoc assembly system able to manufacture the requested product. During production, the modules self-adapt to ensure that assembly continues in degraded modes. For example, they might adapt to each other's speeds, or functioning modules might take over from a faulty one.[7]

Our approach is designed to promote predictability and control in artificial self-organising systems. Predictability is obtained primarily by the dynamic enforcement of policies instantiating the self-organising and resilience mechanisms

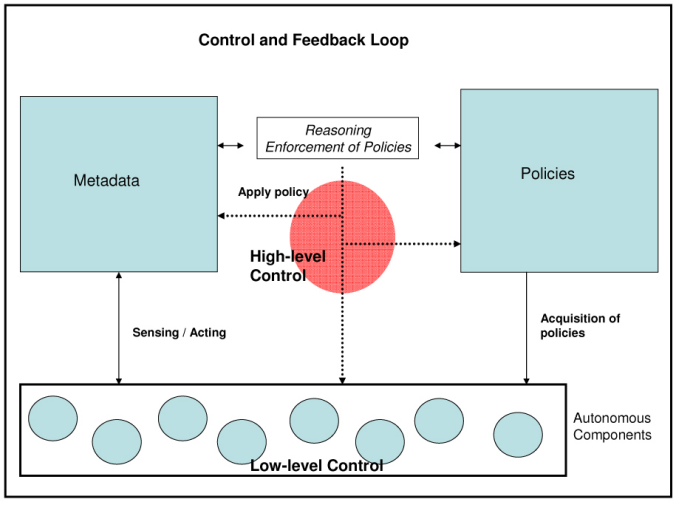

Figure 2. *Control occurs through active modification of metadata, policies and components.*

identified at design-time. Policies turn out to be a useful tool for analysing the emergent properties of the design. The construction of compositional proofs of emergent properties depends on the level of rigour used in the policy and metadata definitions. *Low-level control* (see Figure 2) results from the activity of the components. Components sense and retrieve metadata and policies. Their behaviour causes metadata changes, which in turn cause components to adapt to the new situation. The run-time infrastructure itself is active and, through reasoning services, enforces active (possibly human) *high-level control* by direct reconfiguration of components and modification of the metadata and policies used to drive (change) their behaviour on the fly (see Figure 2). Loose coupling is crucial: policies and metadata are changed without modifying or stopping the components, with new values immediately affecting their behaviour.

As we have seen, self-organising mechanisms are an attractive paradigm for engineering robust artificial systems from simple individual components, but their very flexibility challenges our ability to predict and control their behaviour, and hence their trustworthiness. We have defined a software architecture and established a development method that addresses predictability by exploiting metadata to support decision-making and adaptation based on the dynamic enforcement of explicitly defined policies. Control is obtained by actively modifying metadata, policies or components. Future work will concentrate on enhancing predictability by formal analysis of policies and on

Continued on next page

the spontaneous production of new policies at run-time through reasoning over an internalised model of the self-organising system.

Our work has been conducted in collaboration with Alexander Romanovsky (Newcastle University) and Nicolas Guelfi (University of Luxembourg). The European-Union–funded coordination action PerAda[8] supported travel exchanges between Uninova and Birkbeck College for developing self-organising assembly systems.

Author Information

Giovanna Di Marzo Serugendo
School of Computer Science and Information Systems
Birkbeck College, University of London
London, UK

Giovanna Di Marzo Serugendo is a lecturer. She has a PhD in software engineering from the Swiss Federal Institute of Technology in Lausanne (EPFL). Her research interests are related to the engineering of self-* systems. She co-founded the IEEE International Conference on Self-Adaptive and Self-Organising Systems and is the editor-in-chief of the Association for Computing Machinery's *Transactions on Autonomous and Adaptive Systems*.

John Fitzgerald
School of Computing Science
Newcastle University
Newcastle upon Tyne, UK

John Fitzgerald is a reader whose research concerns formal modelling and its potential to help master the complexity of building provably resilient and fault-tolerant systems, particularly those that are reconfigurable. He currently leads work on resilience in the European Union's Seventh Framework Programme Integrated Project *Deploy*. He has a PhD in computer science from Manchester University, UK, and is chairman of Formal Methods Europe.

References

1. A. Avizienis, J.-C. Laprie, B. Randell, and L. C., *Basic concepts and taxonomy of dependable and secure computing*, **IEEE Trans. on Dependable and Secure Computing 1** (1), pp. 11–33, 2004.
2. J. Kramer and J. Magee, *Self-managed systems: an architectural challenge* **Future of Software Eng. (FOSE'07)**, pp. 259–268, IEEE Computer Society, 2007.
3. G. Picard and M.-P. Gleizes, *The ADELFE methodology - designing adaptive cooperative multi-agent systems* **Methodologies and Software Engineering for Agent Systems: The Agent-oriented Software Engineering Handbook**, pp. 157–176, Kluwer Publishing, 2004.
4. G. Di Marzo Serugendo, J. Fitzgerald, A. Romanovsky, and N. Guelfi, *A metadata-based architectural model for dynamically resilient systems* **ACM Symp. on Appl. Computing (SAC'07)**, pp. 566–572, 2007.
5. G. Di Marzo Serugendo, J. Fitzgerald, A. Romanovsky, and N. Guelfi, *MetaSelf - A Framework for Designing and Controlling Self-Adaptive and Self-Organising Systems* Tech. Rep. BBKCS-08-08, School of Computer Science and Information Systems, Birkbeck College, London, UK, 2008.
6. Y. Chen and A. Romanovsky, *Improving the dependability of Web services integration*, **IT Professional 10** (3), pp. 29–35, 2008.
7. R. Frei, G. Di Marzo Serugendo, and J. Barata, *Designing self-organization for evolvable assembly system* **IEEE Int'l Conf. on Self-Adaptive and Self-Organising Syst. (SASO'08)**, pp. 97–105, IEEE Computer Society, 2008.
8. http://www.perada.eu PerAda web site. Accessed 18 March 2009.